DETAILED ACTION

1. This action is in response to the amendment filed on 08/29/2008.

Claims 1-49, 64-70 are canceled.

Claims 50-63, 71-75 are pending in the application.

Response to Arguments

2. This is in response to the argument remarks filed on 08/29/2008.

In the rejection of Claims 50-63, 71-75, the Office has combined Muller and Peltz. Muller discloses how to uses workflow constructs to create workflow definitions (Figure 1-2, p. 5, A Workflow Definition Tool generates Workflow Definition) that connect to the activities and the operations. The workflow definitions are represented by the activity nodes in which an execution is made when each of these activity nodes is reached (p. 55. sec. 3.2.2). Thus, Java or C++ code would be commonly referred as for workflow language extensions, usually be presented as instance in the execution for these activity nodes. Muller suggested that the programmers can extend a workflow construct with procedural language beyond the abstraction of a workflow (p. 28, sec 2.3). The Office action has addressed: "Muller address a generic application program, Muller does not explicitly address, *program source file includes a source code and classes therein, and workflow definition created using a workflow language that is specified in the form of annotations to the source code and classes*".

Peltz has been used since Peltz discloses, "workflow definition" that is addressed in Muller. With this workflow definition, it uses WSFL ("early work", p. 5) with Java annotation (first paragraph, p. 11), within JavaDoc annotation tags (p. 10) for web development.

Refer to argument remarks that direct the arguments to claims 50 (Previously Presented) (Remarks: p. 7-9) and Claim 75 (Currently Amended) (Remarks, p. 11-12) , and claim 57 (p. 10): The argument remarks submitted that the rejections of independent claims were not obvious based on Muller in view of Peltz. Specifically, Argument remarks contend that Muller and Peltz do not teach a "Web Service source file [that] includes a source code and classes therein and a ***workflow definition created using a workflow language that is specified in the form of annotations to the source code and the classes***, and wherein *said workflow language* extends.

It appears the remarks submitted that annotations normally as not part of the code, therefore it will be ignored in any execution.

It is true; however, the suggestion of Muller using the activity nodes as control tags (p.81) to reach the procedural languages like Java or C++. This teaching addressed the general discussion of this specification: *extending said existing programming language by adding workflow constructs to the existing language*, as it is originally filed. The add of annotation into the claims as an extension is clearly addressed by Peltz ("workflow definition is invoked using WSFL ("early work", p. 5, and p. 10), where WSFL is proposed by IBM for including Java ("early work", p. 5, and first paragraph, p. 11) for performing activities and operations.

It is obvious for combining since Muller teaching provides the architecture of workflow activities, and leaves the activity constructions to developers, while Peltz teaching addresses such

a construction. The use of annotation tag is in connecting to the activities that are left over by Muller as an extension. Thus, the obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 50-63, 71-75 are rejected under 35 U.S.C. 103(a) as being unpatentable by Muller, "Event-Oriented Dynamic Adaptation of Workflows" Model, Architecture, and Implementation", in view of Peltz, "Web Service Orchestration", HP, 1-2003.

As per claim 50: Muller discloses, a workflow definition tool for utilizing a workflow language (i.e. application programs and its application data used as a workflow engine for run-tine (p. 5, and Figure 1-2)). Muller's tool creates a source file (i.e. a box "workflow definition") with workflow constructs to reference to the workflow language and workflow model (See Figure 1-2), which are acting as a workflow execution engine. The workflow language in general is a procedural language such as C++ or Java (p.28), or functional languages LIPS or logic language Prolog. Muller tool execution engine is focused on Procedural Languages.

A Workflow system of Muller discloses (depicting claim 50),

A system for utilizing a workflow language, comprising:

a computer including a processing device operating thereon (Every computer has a CPU/memory);

a program source file stored on a computer readable medium (See Figure 1-2, p. 5, A Workflow Definition Tool generates Workflow Definition), ***wherein the program source file includes a source code and classes therein and workflow definition created using a workflow language that is specified in the form of annotations to the source code and classes , and wherein said workflow language extends the source code with a plurality of workflow constructs*** (See p. 2, "incorporating the process logic directly into the application program". See Figure 1-2, p. 5, An extent of the Workflow Definition to Application Program(s): The Application Program is invoked by Workflow Engine – The Workflow language comprises Java language (Sec. 2.3, p. 28)), ***including constructs for defining parallel processing of a workflow and separate workflow branches therein*** (See Figure 3-5, p. 63), ***and wherein the workflow definition further includes a construct to terminate the parallel processing of the workflow***

when certain conditions are met (See p. 12, Requirement 2, see p. 40, see p. 196, see Figure 5-14 and sec. 5.4.4.3, p. 141, etc)***; and***

means for creating a workflow program according to said workflow definition, including means for the computer to read the source file and process the plurality of workflow constructs to activate a workflow, including creating separate workflow processes corresponding to the separate workflow branches (See Figure 1, p. 5, Workflow Definition Tool),

means for activating each of the separate workflow processes to subsequently generate activities at the computer as defined by each workflow branch (Figure 5-11, p. 130), ***and means for determining when the certain conditions specified in the source file have occurred and then terminating the parallel processing of the workflow*** (See Figure 5-14 and sec. 5.4.4.3).

Muller address a generic application program, Muller does not explicitly address, *program source file includes a source code and classes therein, and workflow definition created using a workflow language that is specified in the form of annotations to the source code and classes*

Peltz discloses "workflow definition is invoked using WSFL (See its "early work", p. 5), where WSFL includes Java (See its first paragraph, p. 11), i.e. program source file that includes source code and classes therein, and JavaDoc annotation tags (See its p. 10) in the workflow language that is specified in the form of annotations to the source code and classes.

It is obvious to the ordinary in the art at the time to combine or to include program source file as Java Application because Java is commonly used by every developer for developing web

and business applications, where it is an object-oriented program having source code and classes therein. Furthermore, using Java Application as of Peltz in the Muller' Applications only yields the same and predictable results, and it is obvious for every ordinary in the art to include it in the teaching of Muller.

As per claim 51: Incorporated to the rejection of Claim 50, with regard to ***The system of claim 50, wherein the workflow definition is invoked by a executing a software application*** (See Peltz: "workflow definition is invoked using WSFL" (See its "early work", p. 5), where WSFL includes Java (See its first paragraph, p. 11), or JavaDoc annotation tags (See its p. 10)).

As per claim 52: Incorporated to the rejection of Claim 50, with regard to The ***system of claim 50, wherein the plurality of workflow definition constructs are provided as XML commands*** (Muller: See p. 314, 5. rule definitions in the source file include XML –Integration) ***that are then used as annotations to the source code and classes*** (See Peltz: p. 10).

Peltz discloses workflow definition files are used for web service (XML-Based Language), and constructs in the files are XML-commands (See its listings, e.g. p. 7, p.8, etc).

It is obvious to the ordinary in the art at the time to combine or provide XML-commands because XML is an open source used for Web-service. Furthermore, using XML-commands in a workflow definition file yields the same and predictable results, and it is obvious for every ordinary in the art to include it in the teaching of Muller for web purposes.

As per claim 53: With regards to, ***The system of claim 50, further comprising a light-weight virtual machine at the computer that processes the workflow and that is enabled to, at a particular point in the workflow process, save the workflow's execution space including***

program stack and variable state, and, at a later point in time, revive the workflow at the same point in the workflow process using the saved program stack and variable state.

It should be notice that virtual machine is only a browser using web-service. The Reference of Peltz discloses such of web service. It should be note that Java is a stack-based application program developed by Sun Microsystems,

Therefore, it is obvious to ordinary in the art for knowing that the further claim merely recites the rules, the techniques, and the principles, which have been already developed by others as requirement in the Web, and included in the process of workflows as of Muller or Peltz for conforming to the requirements of the Web.

As per claim 54: With regards to,

The system of claim 50, wherein the program source file is a Web Service file that includes the workflow definition constructs. Refer to Peltz: The program with workflow definition constructs disclosed in Peltz is Java Web Service file.

As per claim 55: With regards to, ***The system of claim 54, wherein the workflow definition constructs of the Web Service file also references Java methods and variables for a software application running on the system and using the workflow.***

The workflows disclosed in Peltz is Java Web Service file using method and variables running on a computer system using workflow – For example see Figure 5, p. 10.

As per claim 56: With regards to, ***The system of claim 54, wherein workflow definitions are provided as a separate Work Flow file that includes workflow definition commands, and that***

are invoked by the Web Service file using the workflow definition constructs, to use the workflow as defined by the Work Flow file.

The Workflow definition(s) in Muller figure 1-2 is provided as a separate Work Flow file that includes workflow definition commands, and that are invoked by the Applications program(s) using the workflow definition constructs in the files, to use the workflow as defined by the Application Program(s) as seen in Muller Figure 1-2,

It is obvious to the ordinary in the art to include separate Java Work Flow file as seen in the reference of Peltz because it yields predictable results.

As per Claim 71: Regarding, ***The system of claim 55, wherein the Web Service file includes the workflow definition constructs as a plurality of XML workflow annotations to the source code and classes defined in the Web Service file***. See Peltz: p. 10.

As per Claim 72: Regarding, ***The system of claim 71, wherein the XML workflow annotations to the source code and classes define a flow logic that can then reference the methods and variables defined in the Web Service file***. See Peltz: p. 10.

As Per Claims 57-63, and 73-74: The rejection of the claims is the same as of Claims 50-56, and 71-72. Refer to the rationale as addressed in the rejection of Claims 51-56, and 71-72 above.

As Per Claim 75: The rejection of the claim is the same as of Claim 50. Refer to the rationale as addressed in the rejection of Claim 50 above.

Conclusion

5. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (571) 272-3706. The examiner can normally be reached on 8:00AM to 4:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708.

The facsimile number for the organization where this application or proceeding is assigned is the Central Facsimile number **571-273-8300**.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system.

Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TTV
December 05, 2008

/Ted T. Vo/
Primary Examiner, Art Unit 2191